Universiteit Leiden

Master Computer Science

Optimizing the linear interpolation of sparse and dense retrieval models

Name: Nan Sai
Student ID: s3135667

Date: [dd/mm/yyyy]

Specialisation: Computer Science: Data Science

1st supervisor: Suzan Verberne
2nd supervisor: Zhaochun Ren
additional supervisor: Amin Abolghasemi

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Interpolating sparse and dense retrieval models has been shown to be an improvement for document retrieval tasks compared to using them alone. In previous work, the interpolation coefficients for queries in the entire dataset took the same value, but experiments showed that the optimal coefficients were different for each query, and greater improvements were obtained when each query was interpolated using its own optimal coefficients. How to determine the optimal coefficient for each query, therefore, remains a problem to be decided.

In this thesis, we aim to devise a pipeline to improve retrieval performance by training a predictor to estimate a linear interpolation coefficient for each query. We use a cross-encoder with a head of a transformers-based deep language model to predict the coefficients. Our approach involves considering different input formats, such as individual queries, concatenating queries with passages from ranked result lists, and query expansions. In addition, we investigate the impact of interpolating different sparse and dense retrieval models and different depths of evaluation metrics.

Contents

Chapter 1

Introduction

Information retrieval (IR) is the process of finding and retrieving relevant information from a collection of unstructured or structured data sources such as text documents, images, videos, or databases. Information retrieval is mainly concerned with finding documents related to the query. The relevance measurement between query and document is the core issue of information retrieval, focusing on the accuracy of retrieval results. A common way is to estimate the relevance score of each document for a given query and then sort according to this, usually taking the top 1000 (Craswell et al., 2023) 10. (2022) as the final ranking lists.

Traditional sparse retrieval models estimate relevance based on overlapping words in queries and documents, on which the weights are modeled and the scores of the different weights are combined to obtain the relevance score. For example, BM25 (Robertson and Zaragoza, 2009) obtains a relevance score by combining word frequency, document length, and inverse document frequency. Another category of models are the dense retrieval models (Karpukhin et al., 2020), which map the input text (query and document) to an independent dense representation (semantic representation vectors) and calculate the spatial similarity between them using an approximate nearest neighbor algorithm for efficient retrieval. Examples of popular dense retrieval models are ANCE (Xiong et al., 2020), TCT-ColBERT-V2 (Lin et al., 2021), DistilBERT KD

(Hofstätter et al., 2021), and SBERT (Reimers and Gurevych, 2019). Each of these model types aims to capture different aspects of relevance. Dense retrieval models are made for matching similarities between queries at a semantic level, rather than word-based matching, such as BM25.

Although the dense retrieval models have shown to be very effective for document retrieval, a decision needs to be made as to whether the scores output from dense retrieval models need to be integrated with those from sparse retrieval models (as shown in equation 1.1, where α is referred to the weighted interpolation coefficient), as sparse retrieval models may capture some exact term matching information that is ignored by dense retrieval models. BERT re-ranker is referred to as an approach using BERT to re-rank top-k passages retrieved by bag-of-words retrievers such as BM25. Lin et al. (2021) claim that it is unnecessary to interpolate exact terms matching scores (BM25) and BERT scores because BERT has already captured all useful relevant signals of BM25. But other studies report that interpolating the sparse and other novel BERT-based dense retrievers results could increase the retrieval performance because they capture different relevance signals (Askari et al., 2023; Lin and Ma, 2021; Lin et al., 2021, 2020). Wang et al. (2021) investigate combining relevance signals from sparse retrievers with those from dense retrievers via linear interpolation and showed that the interpolation results in higher retrieval performance.

$$s(p) = \alpha s_{sparse}(p) + (1 - \alpha) s_{dense}(p) \tag{1.1}$$

They reported that when the interpolation coefficient $\alpha = 0.3$, the interpolation of ANCE and BM25 achieved higher nDCG@10 scores on **MS MARCO** dataset than only using ANCE or BM25 models. However, for each query, the linear interpolation coefficient that reaches the highest nDCG@10 is not always 0.3. A higher nDCG@10 is achieved when each query has its own optimal α value, which is called the oracle value (Wang et al., 2021). But there is not a good way to estimate it. Hence, there is still a great possibility of improvement between the best value and the oracle value. Additionally, as we will show in our result section, different test sets on the same collection do not necessarily

have the same optimized interpolation coefficient.

In this thesis, we provide a thorough investigation of whether a pipeline can be devised to improve retrieval performance by obtaining a linear interpolation coefficient corresponding to each query. We perform different experimental methodologies to estimate the interpolation coefficient in the combination of sparse and dense retrieval models. Our estimator is based on BERT Devlin et al. (2019) and DistilRoBERTa-base (Sanh et al., 2019) as widely-used pre-trained language models with high contextualization power. We propose to use a large pre-trained transformers-based deep language model as a cross-encoder which performs full attention over the input pair, to predict the corresponding coefficients for each query. The next question is what information to use as input to the pre-trained language model. The simplest is to use the query directly as input, with the optimal linear interpolation coefficient corresponding to each query as the label to train the model. For a more sophisticated approach, we can combine information from the query and ranked result lists as input. There is a very brutal approach: directly concatenating the input query and the top-k retrieved documents from the ranked result lists of sparse and dense retrieval models. Another approach to combine information from queries and ranking lists is from the pipeline of the pseudo-relevance feedback RM3 (Abduljaleel et al., 2004; Lv and Zhai, 2010). In this process, the RM3 relevance model is used to combine the queries and document information from ranking lists obtained in the first round of retrieval, which is also called query expansion.

The same approach can also be used to combine queries and information from documents in the sparse and dense retrieval ranking lists. I used the above methods to combine each query with the information from sparse and dense retrieval ranking lists separately. The extended queries are created by combining the original queries and the information from sparse and dense retrieval ranking lists respectively. Next, I use the two extended queries of the same query to predict the linear interpolation coefficients. Now that we have a set of predefined sentence pairs that have been scored, we train a cross-encoder

(Reimers and Gurevych, 2019) to predict the linear interpolation coefficients for these sentence pairs.

The research questions that we address in this paper are:

- **RQ1**: How to effectively estimate the interpolation coefficient in hybrid ranking with sparse and dense retrieval models? We use large pre-trained language models to predict the interpolation coefficient α for each query. For the inputs to the model, several different approaches have been proposed: query alone, concatenating query with top-k documents, and query expansion.
- **RQ2**: How do the results of interpolation coefficient estimation vary for different evaluation metrics?
- **RQ3**: How do the results of interpolation coefficient estimation vary across different sparse and dense retrieval models?

My research thesis is divided into six chapters. In Chapter 2, I provide the background information. In Chapter 3, the experimental methodology used in the thesis is presented in detail. In Chapter 4, I show all the experiments and results and analysis of them. The discussion is presented in Chapter 5. And the conclusion is given in Chapter 6.

Chapter 2

Background

This chapter covers the theoretical foundations of information-retrieval-related knowledge and related work on the interpolation of sparse and dense retrieval models. Section 2.1 and 2.2 introduce the basics of sparse and dense retrieval and the sparse and dense retrieval models used in chapter 4. Section 2.3 introduces some work on the interpolation of sparse and dense retrieval models. In section 2.4, we provide background on the query expansion method which we use to reformulate the query by adding the information from the retrieved ranked lists. In section 2.5, we demonstrate the basic concept of the cross-encoder.

2.1 Sparse and Dense Retrieval

In terms of the type of representation and indexing method retrieval models can be divided into two categories (Fan et al., 2022): Sparse Retrieval: improves retrieval efficiency by obtaining sparse document representations based on words and building inverted indexes. Dense Retrieval: do retrieval by mapping the input text (query and document) to independent dense representations. The following is the introduction of dense and sparse retrieval respectively.

- Typical sparse models include TF-IDF and BM25 (Robertson and Zaragoza, 2009), which can efficiently match keywords through inverted indexes. Such methods can be viewed as representing questions and paragraphs

as high-dimensional, sparse vectors. The sparse vector representation works well for literal matching, but not for semantic matching.

- Dense Retrieval (Zhao et al., 2022) directly changes the original retrieval mode, maps queries, and documents into semantic space, and then uses the ANN algorithm for retrieval. Generally, the two-tower encoding method is used to encode the query and the document separately to obtain independent expressions of the two, so that the queries can be indexed "on-the-fly" (Zhuang and Zuccon, 2021). Examples of popular dense retrieval models are ANCE (Xiong et al., 2020), TCT-ColBERT-V2 (Lin et al., 2021), DistilBERT KD (Hofstätter et al., 2021), and SBERT (Reimers and Gurevych, 2019), which are all used in the following experiments.

2.2 Sparse and Dense Retrieval Models Used

Before performing linear interpolation of sparse and sense retrieval, the first round of retrieval is performed using the sparse and sense retrieval models respectively to obtain the ranking lists of top 1000 passages.

2.2.1 BM25

BM25 is the most dominant algorithm for computing query-to-document similarity scores in information indexing. BM stands for Best Match and 25 refers to the 25th iteration of the algorithm (Robertson and Zaragoza, 2009).

BM25 has the same main components as TF-IDF, the BM25 formula consists of three main components:

- Relevance between each word q_i in the query and the document d
- Relevance of each word q_i in the query to the query (only used if the query is too long)
- The weight of each word

The general formula for BM25:

$$Score(Q, D) = \sum_{i}^{n} W_i R(q_i, d) \tag{2.1}$$

$$Score(Q, D) = \sum_{i}^{n} W_i R(q_i, d) \tag{2.2}$$

where Q is a query, q_i is the word in the query, and d is a search document. The design of BM25 is based on an important finding: the relationship between word frequency and relevance is non-linear, i.e. the relevance score of each word to a document does not exceed a specific threshold, and its impact does not increase linearly once the number of occurrences of the word reaches a threshold that would be related to the document itself. Thus, when portraying word-document similarity, BM25 is designed in such a way that:

$$S(q_i, d) = \frac{(k_1 + 1)tf_{td}}{K + tf_{td}} \tag{2.3}$$

$$K = k_1(1 - b + b * \frac{L_d}{L_{ave}}) \tag{2.4}$$

where tf_{td} is the word frequency of word t in a document d, L_d is the length of document d, L_{ave} is the average length of all documents, and the variable k_1 is a positive parameter used to normalize the range of word frequencies in the article; when $k_1 = 0$, it is a binary model and a larger value corresponds to the use of more primitive word frequency information. b is another adjustable parameter ($0 < b < 1$), which determines the range of information content using the document length: when $b = 1$, the document length is fully used to weigh the words, and when $b = 0$, the document length is not used.

When the query is very long, we also need to carve out the weight between the words and the query. For short queries, this item is not necessary.

$$S(q_i, Q) = \frac{(k_3 + 1)tf_{tq}}{k3 + tf_{tq}} \tag{2.5}$$

where tf_{tq} indicates the word frequency of word t in the query, and k_3 is an adjustable positive parameter to correct for the range of word frequencies in the query. The final formula for BM25 is:

$$RSV_d = \sum_{t \in q} [log \frac{N}{df_t}] \cdot \frac{(k_1 + 1) tf_{td}}{k_1((1 - b) + b \times (\frac{L_d}{L_{ave}})) + tf_{td}} \cdot \frac{(k_3 + 1) tf_{tq}}{k_3 + tf_{tq}} \tag{2.6}$$

BM25 often fails to retrieve documents on the top of the ranking list but has a high recall value.

2.2.2 ANCE

ANCE is a very effective dense retrieval model. To solve the problem of not providing much information because of the excessive use of simple negative examples (random or in-batch negative sampling), an Approximate nearest neighbor Negative Contrastive Estimation (ANCE) is proposed, i.e. a KNN-like method to find the nearest negative classes that contribute most to contrastive learning. Specifically, a RoBERTa-base pre-trained model is used to initialize the dual tower model. Then a warm-up is first done with BM25 (hard negative case sampling with BM25), after which the index is updated by an asynchronous method, using the checkpoint of the model being trained for hard negative case sampling. (Xiong et al., 2020)

When building the algorithm, it is important to consider how to align the data distribution during training and testing, i.e. how we learn it asynchronously. ANCE uses the standard DR model and loss function:

$$f(q, d) = BERT\text{-}Siamese(q, d) \tag{2.7}$$

$$l(q, d^+, D^-) = NLL(q, d^+, D^-) \tag{2.8}$$

The only difference is the negative sample during training:

$$D^- = D^-_{ANCE} = ANN_{f(q,d)} \backslash D^+ \tag{2.9}$$

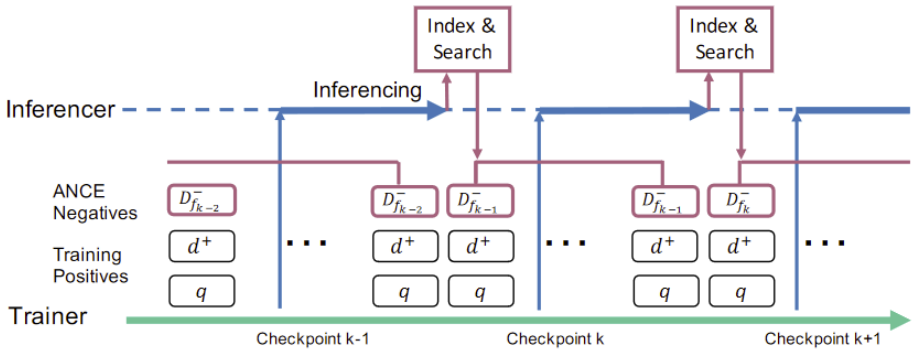


Figure 2.1: ANCE Asynchronous Training. (Xiong et al., 2020)

The ANN searches out the index using the learned representation model $f()$, which makes the inference the same at the time of inference as at the time of training, eliminating the difference in data distribution between these. Because the training process is random, encoder f will be updated at every step. To update the negative samples of ANCE the following two steps are needed:

- inference: update the representation of all documents with the new encoder
- index: use the updated representation to reconstruct the ANN index

So ANCE only refactors the ANN index after every k checkpoints.

2.2.3 SBERT

BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) have achieved state-of-the-art performance on the Semantic Textual Semilarity (STS) benchmark (Cer et al., 2017), using a cross-encoder structure: two sentences are spliced and fed into the model, and a transformer network with self-attention is used to obtain the final prediction. However, they both require two sentences to be fed into the network at the same time, which leads to a huge computational overhead. Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) is proposed to address the huge time overhead of BERT semantic similarity and the fact that its sentence representations are not suitable for unsupervised

tasks such as clustering, sentence similarity calculation, etc. Sentence-BERT (SBERT) is a dual network based on pre-trained BERTs that can obtain semantically meaningful chapter vectors. Sentence-BERT uses a forensic dual network structure to obtain a vector representation of sentence pairs and then pre-trains the similarity model. The network structure generates semantically meaningful embedding vectors V_q V_d of query and document separately, and the embedding vectors V_q and V_d are closer if they are semantically similar. So that they can be used for similarity calculations (cosine similarity, Manhattan distance, Euclidean distance), and then ranked and retrieved according to the similarity scores obtained. Since it is no longer necessary to put each pair of query and document into the model for computation, but rather the query and document embedding vectors (the vector of documents that can even be indexed in advance) are generated separately, the retrieval efficiency is greatly improved.

SBERT adds a Pooling operation to the output of BERT/RoBERTa, calculating the average of all Token output vectors as the whole sentence vector. In order to be able to fine-tune BERT/RoBERTa, Siamese and Triplet Network (Schroff et al., 2015) were used to update the parameters to achieve a more semantically informative generated sentence vector.

2.2.4 TCT-ColBERTv2

Tightly-Coupled Teacher ColBERT(TCT-ColBERTv2) (Lin et al., 2021) is an improved version of the ColBERT (Khattab and Zaharia, 2020) model. ColBERT introduces a late-interaction architecture that uses BERT to encode query and document separately, followed by fine-grained modeling of the two correlations using lightweight and efficient modules. By delaying the query-document interaction, ColBERT is able to gain both the encoding power of BERT and the ability to compute the document representation in advance, which speeds up the processing of online queries. The "late interaction" is also a dual tower architecture, encoding query and document separately, and then taking the encoded vector representation and using some computational func-

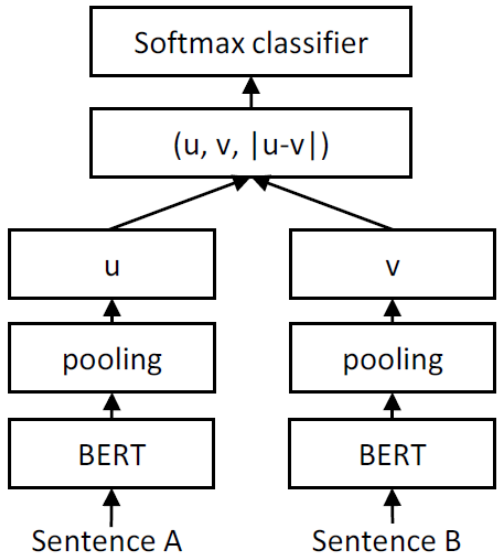


Figure 2.2: SBERT architecture. (Reimers and Gurevych, 2019)

tion (e.g. MaxSim) to compute the interaction between query and document separately, depicting token similarity.

The key innovation of TCT-ColBERT is the use of an in-batch knowledge distillation with a tightly-coupled teacher-student architecture in the training process. The teacher model is a larger, more powerful version of the student model, and it is used to generate targets for the student model during training. By using targets, the student model is able to learn from the more informative signals provided by the teacher model, resulting in better retrieval performance. In the in-batch distillation process, it only exploits all possible query–passage triplets within a minibatch instead of all combinations, which is also called tight coupling.

2.2.5 DistilBERT KD

The most effective BERT-based (Devlin et al., 2019) neural ranking models are called BERT_{CAT}. Both query and document are fed into the pre-trained model, and interaction computing is conducted at each layer of the entire

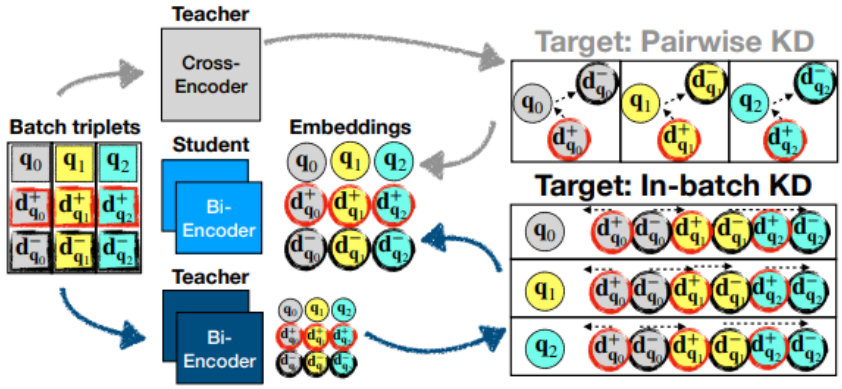


Figure 2.3: Illustration of the differences between pairwise knowledge distillation and in-batch knowledge distillation. (Lin et al., 2021)

neural network through the attention mechanism. DistilBERT KD is a type of knowledge distillation model (Hofstätter et al., 2021). The student model is taught by the state-of-the-art full interaction BERT$_{CAT}$ model by using cross-architecture knowledge distillation and leads to improved effectiveness. Instead of optimizing the raw scores, the margin between a pair of relevant and non-relevant passages with a Margin-MSE loss is used, and this method outperforms a simple pointwise MSE loss:

$$\begin{aligned}\mathcal{L}(Q, P^+, P^-) = MSE(M_s(Q, P^+) - M_s(Q, P^-),\\ M_t(Q, P^+) - M_t(Q, P^-))\end{aligned} \tag{2.10}$$

where Q is queries, P^+ is relevant passages, P^- is non-relevant passages, M_t is teacher model and M_s is student model.

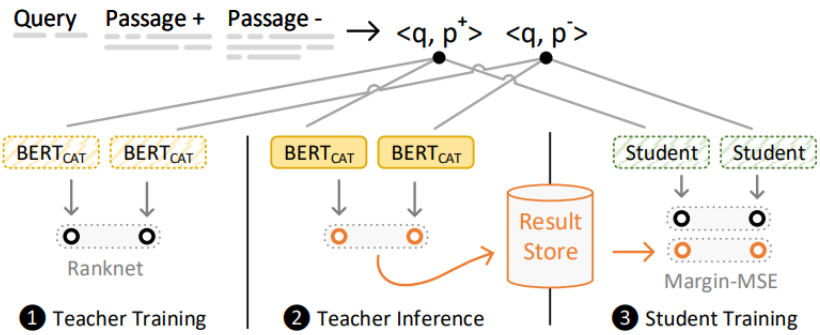


Figure 2.4: The knowledge distillation training process of DistilBERT KD. (Hofstätter et al., 2021)

2.3 Interpolation of Sparse and Dense Retrieval Models

Using the sparse and dense retrieval models, each passage is given a score to indicate its relevance to the query. The ranking lists are obtained by sorting the passages according to their scores. An implementation of interpolating sparse and dense retrievers is linear interpolation, as shown in equation 2.11:

$$s(p) = \alpha s_{sparse}(p) + (1 - \alpha) s_{dense}(p) \tag{2.11}$$

where $s_{sparse}(p)$ is the normalize sparse retrieval score of passage p, $s_{dense}(p)$ is the normalize dense retrieval score of passage p and the parameter α is the linear interpolation coefficient. α controls the relevance weight of sparse and dense retrieval scores. Wang et al. (2021) further thoroughly investigated the importance of interpolating dense retrieval and BM25 scores. They depict the MRR@10 and nDCG@10 interpolation results between different dense retrievers and BM25 on **MS MARCO dev** (Bajaj et al., 2016), **TREC 19** (Craswell et al., 2020) and **TREC 20** (Craswell et al., 2021) dataset. In **MS MARCO dev** dataset, each query corresponds on average to only one relevant passage. And **TREC 19** and **TREC 20** use deep judgment pools and graded relevance

labels (relevance label = level 3,2,1,0). It is clear that the highest effectiveness is not obtained when using dense retrievers or BM25 alone, but rather the higher effect is obtained after interpolating dense retrievers and BM25. For example, the best MRR@10 value is obtained at $\alpha = 0.3$ when interpolating RepBERT and BM25 instead of using each of them alone. When considering the other cases as well, they find that the best results are not always obtained at $\alpha = 0.3$. For example, the best nDCG@10 value is obtained at $\alpha = 0.3$ when interpolating ANCE and BM25. Then, they discuss the interpolation results on deep evaluation metrics. They find that the improvement of interpolating dense retrieval and BM25 scores is much more significant on deep evaluation metrics (evaluation metrics for considering more top passages in ranked lists like nDCG@1000, MAP) than on shallow evaluation metrics (evaluation metrics for considering fewer top passages in ranked lists like nDCG@10). They point out that dense retrievers are very effective at encoding passages with strong relevance signals(relevance label = level 3) and do a better job at this than BM25. However, when it comes to modeling weaker relevance signals, it is instead the BM25 that performs better. Interpolating them can be a good way to make up for their weaknesses. They also discuss the oracle values. More significant gain is obtained when the most efficient interpolation coefficient α corresponding to each query is used to obtain the retrieval result.

Li et al. (2022) investigate the interpolation of dense and sparse retrieval results in the context of Vector-pseudo-relevant feedback (VPRF). VPRF (Li et al., 2022) includes two round dense retrievals. VPRF method gets the top retrieved passages' dense vectors in the first round retrieval and then uses these vectors to enhance the original query's dense representation. And these enhanced query's dense representations are used for the second round retrieval. It is clear that both rounds of retrieval can be interpolated with sparse retrieval. They explored three cases: interpolation in the first round only, interpolation in the second round only, and interpolation in both rounds. Two sparse retrievers (uniCOIL (Lin and Ma, 2021) and BM25 (Robertson and Zaragoza, 2009)) and three dense retrievers (ANCE (Xiong et al., 2020), TCT-ColBERT-V2 (Lin et al., 2021), and DistilBERT KD (Hofstätter et al., 2021)) are considered to be used for interpolation. In cases of interpolation with BM25, conducting in-

terpolation in both rounds of retrievals achieves the highest effectiveness most of the time and across all three dense retrievers and **TREC 19**, **TREC 20** datasets. In cases of interpolation with uniCOIL, conducting interpolation in both rounds of retrievals also show high results in most of the time. Furthermore, only interpolating with uniCOIL in the second round of retrieval shows high effectiveness, not BM25. Then they further investigate the difference in performance between learned sparse retriever: uniCOIL and unsupervised sparse retriever: BM25. Interpolation with BM25 tends to have relatively high recall while interpolation with BM25 uniCOIL tends to have higher nDCG@10 and MAP.

2.4 Query Expansion

Query expansion is a process in information retrieval that involves selecting and adding terms to a query aiming at minimizing query-document mismatches and improving retrieval performance (Vechtomova and Wang, 2006). One of the most common methods is to use the RM3 relevance models for query expansion. This is often used between the first and second retrieval steps of pseudo-relevance feedback (Abduljaleel et al., 2004; Lv and Zhai, 2010). In the field of information retrieval, relevant feedback is to make use of the initial results returned for a given query, and whether these results are relevant to the new query or not. Pseudo-relevant feedback provides an automatic local analysis method. This approach starts by finding an initial result from the most relevant documents through a general retrieval and then assumes that the top k ranked documents are relevant.

It combines query and relevant information of top k documents obtained in the first retrieval step. The RM3 relevance model first retrieves a set of top-ranked documents based on the original query. It then estimates a new query model by combining the original query with the language model probabilities of the top-ranked documents. The new query model is then used to retrieve a final set of documents. So we can use the same method for query expansion purposes.

2.4.1 RM3 Relevance Model

In the absence of feedback information, the MLE method is often used to estimate the query language models:

$$p(\omega \mid Q) = \frac{c(\omega, Q)}{|Q|} \tag{2.12}$$

where $c(\omega, Q)$ is the occurences of word ω in the query Q, and $|Q|$ is the total number of words in the query Q. In order to take the feedback information into account (assume the top k retrieve documents in the first stage retrieval are all relevant), we have to estimate more accurate query language models. There are some effective methods for query model estimation based on pseudo-feedback techniques. The formula of the first estimation method of relevance model (RM1) (Lavrenko and Croft, 2001) is:

$$p_1(\omega \mid Q) \propto \sum_{\theta_D \in \Theta} p(\omega \mid \theta_D) p(\theta_D) \prod_{i=1}^{m} p(q_i \mid \theta_D) \tag{2.13}$$

where Θ represents the set of smoothed document models in the pseudo feedback collection(top k documents) and Q represents the set of all queries. The RM1 relevance model $p_1(\omega \mid Q)$ can be interpolated with the original query language model $p(\omega \mid Q)$ to further improve the performance (Lv and Zhai, 2009). And the interpolated relevance model is called the RM3 relevance model:

$$RM3 : p(\omega \mid \theta'_Q) = (1 - \alpha) p(\omega \mid \theta_Q) + \alpha p_1(\omega \mid \theta_Q) \tag{2.14}$$

2.5 Cross-Encoder

Cross-Encoder is a paradigm of scoring pairs of sentences that perform full self-attention over the pair. This model is usually built based on a Transformer-based language model (such as BERT or RoBERTa). The cross-encoder concatenates two sentences together with the separator $[SEP]$ and feeds them into a language model. Equation 2.15 shows the cross-encoder input formulation. The $[CLS]$ token is placed at the beginning of the first sentence and the rep-

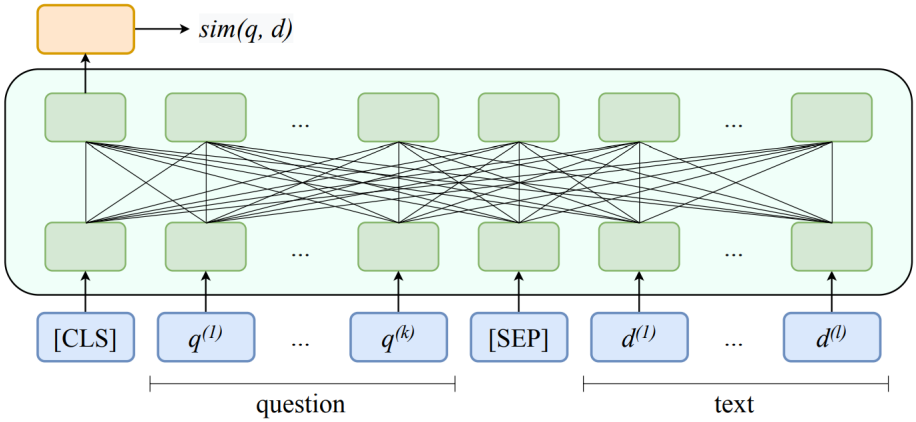


Figure 2.5: Cross-Encoder architecture (Zhao et al., 2022)

resentation vector C obtained by the Transformer-based language model can be used for subsequent tasks. The $[SEP]$ token is used to separate two input sentences, e.g. input sentences A and B. The $[SEP]$ flag is added between sentences A and B.

$$[CLS] \quad SentenceA \quad [SEP] \quad SentenceB \tag{2.15}$$

Equation 2.16 shows the first output of the transformer which is the embedding vector of the interactions between the sentences pair:

$$y_{SentenceA,SentenceB} = first(T(SentenceA, SentenceB)) \tag{2.16}$$

At the top of the language model, there is a classification or relevance scoring head that is trained to predict a target score as shown in equation 2.17, where W is the classification or relevance scoring layer.

$$score(SentenceA, SentenceB) = y_{SentenceA,SentenceB}W \tag{2.17}$$

However, cross-encoders need to compute a new encoding for each pair of input sentences, which results in high computational overhead in cases with massive pairs of sentences.

Chapter 3

Methodology

Although Wang et al. (2021)'s work demonstrates that interpolating sparse and dense retrievers yields improvement compared to using them alone, the improvement is larger when each query has its own interpolation coefficient (α). The retrieval result obtained when the α value used in the interpolation is the ideal α value for each query is what we call the **oracle value**. There is still a significant gap between the oracle values and global coefficient values. So we wanted to devise a process to train a model that could predict the optimal α of queries, thus enabling a further improvement in retrieval efficiency. In this thesis, I choose BM25 (Robertson and Zaragoza, 2009) as the sparse retrieval model and ANCE (Xiong et al., 2020), SBERT (Reimers and Gurevych, 2019), TCT-ColBERTv2 (Lin et al., 2021), DistilBERT KD (Hofstätter et al., 2021) as the dense retrieval models. In the process of using them for retrieval, they are used as encoders to encode passages and queries respectively, and then post-interactively compute queries and passages for ranking passages. If I change the queries set several times while the passages set remains the same, only performing the "on-the-fly" query encoding is necessary, which can greatly reduce the amount of GPU computation. To generate these retrievers' results to be used for interpolation, I use the implementation provided by Pyserini (Lin et al., 2021), which already has some pre-built index of some benchmark datasets. The linear interpolation coefficient is defined as α. The ranking list is defined as L. L_{dense}, L_{sparse} represent ranking lists obtained by dense and

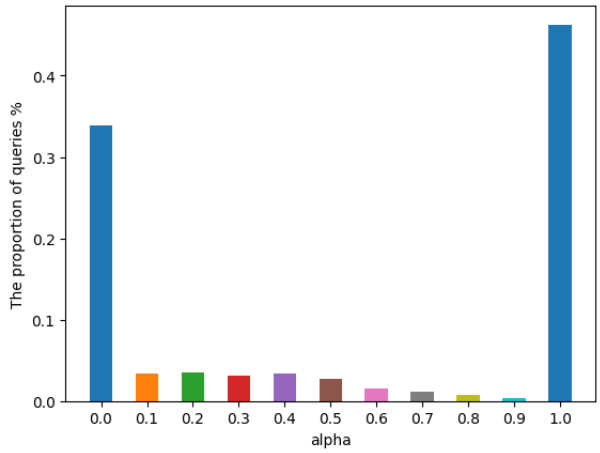


Figure 3.1: MS MARCO dev: The optimal alpha distribution for each query obtained according to the nDCG@10 value.

sparse retrieval models, and $L^{1:k}$ represents top k passages retrieved in the ranking list.

First, we obtain the optimal α for each query. Then, we utilize it as the label to estimate α in two different methods: query-based coefficient estimation and reformulation-based coefficient estimation.

3.1 Obtaining optimal α for each query

As shown in Figure 3.1, the best α value is different for each query. By using the next method, we can calculate the optimal interpolation coefficient for each query. As shown in Figure 3.2, for a query, the dense and sparse retriever return two ranking lists L_{dense} L_{sparse}separately which contain retrieved passages and their corresponding relevance scores (normally containing the top 1000 rankings). The sparse and dense ranking lists are then linearly interpolated according to equation 3.1. In this way, each passage is given a new fused relevance score and re-sort them from highest to lowest to obtain a new ranking list (also containing the top 1000 rankings). α takes eleven values from $[0.0, 0.1, 0.2.....0.9, 1.0]$ respectively. The fused ranking lists are evaluated using evaluation metrics (such as nDCG@10), and the alpha value corresponding

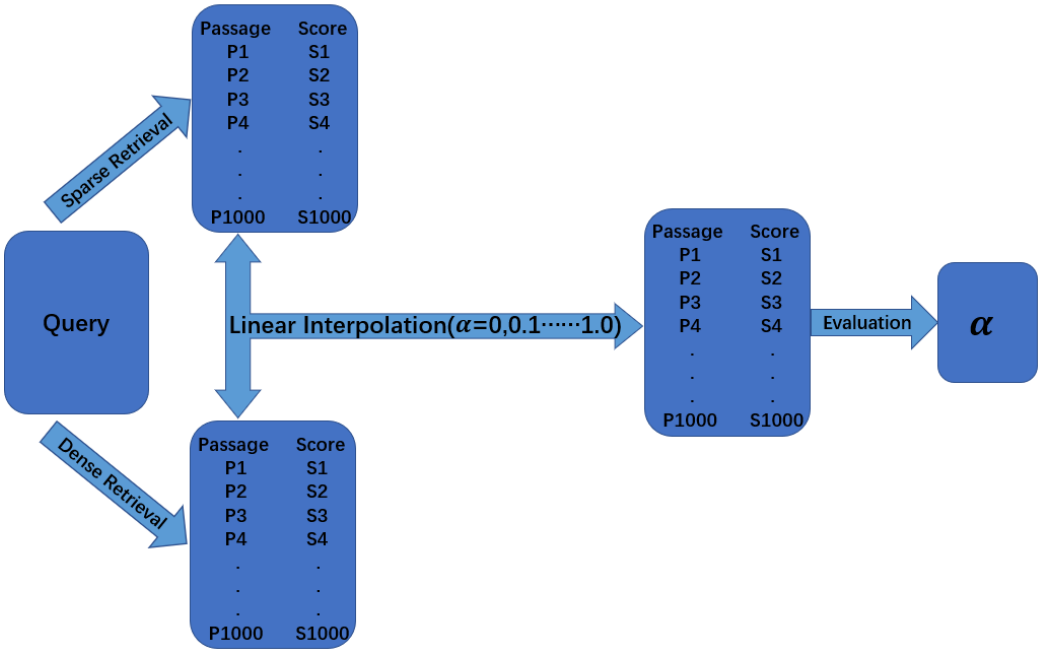


Figure 3.2: The process of obtaining optimal α for each query

to the fused ranking with the best retrieval effect is selected as the optimal α.

$$s(q,p) = \alpha s_{sparse}(q,p) + (1-\alpha)s_{dense}(q,p) \tag{3.1}$$

3.2 Query-based Coefficient Estimation

First I consider trying to use only the information in the query to predict its corresponding optimal α value. As shown in Figure 3.3, a query is first fed into the BERT-like language model, after which a score is obtained through the linear layer. The optimal α values obtained previously are used as labels to fine-tune the neural network model so that it can predict the α of the queries.

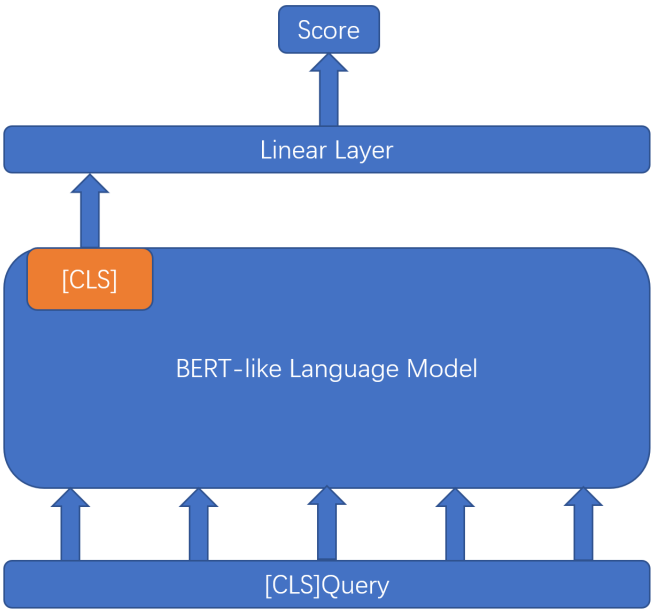


Figure 3.3: The pipeline of using one query.

3.3 Reformulation-based Coefficient Estimation

3.3.1 Adding Information of Ranking List to Query

In this thesis, we propose two ways to combine the query with the information in its corresponding ranking list:

Concatenating Query and $L^{1:k}$ Documents

The first is a brutal approach: directly concatenating the input query and the $L^{1:k}$ retrieved documents from ranking lists.

$$new\_query = query + passage_1 + passage_2 + ... + passage_k \tag{3.2}$$

The drawback of this approach is that the common language models nowadays have input limits, e.g. BERT has a maximum of 510 tokens, and when the length of the passages is too long, the amount of information that can be added becomes very limited.

Query Expansion

The second approach is to use the query expansion method for combining. First, assume that the $L^{1:k}$ passages in the ranking list are all relevant (feedback documents). Then using the RM3 relevance model returns k terms to be added after the query.

$$query \xrightarrow{RM3} query + term_1 + term_2 + ... + term_k \tag{3.3}$$

This method uses fewer terms but combines more information than the first method. To conduct the query expansion, I use the RM3 method provided by PyTerrier (Macdonald and Tonellotto, 2020).

3.3.2 Training Cross-Encoder

Since the cross-encoder requires a set of predefined sentence pairs to be scored as input, I choose a query combined with sparse and dense retrieval ranking lists respectively as input. As shown in Figure 3.4, two sentences are concatenated using a special token $[SEP]$ to separate them, which is then put into the BERT-like model. Finally, a score is output after a linear layer. The optimal alpha value can be used as a label during the training process, and the final trained model is expected to predict the alpha corresponding to each query, thus improving the retrieval performance.

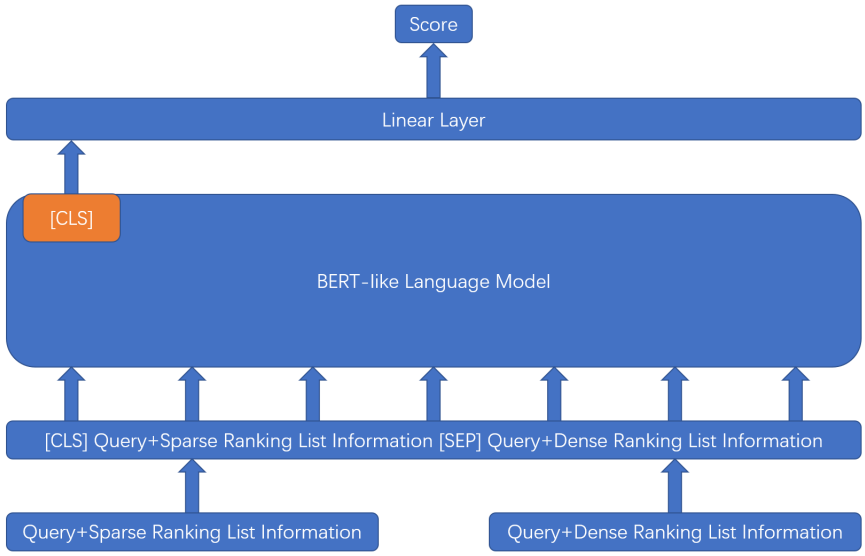


Figure 3.4: Structure of cross-encoder

Chapter 4

Experiments and Results

In this chapter, we first present the passage sets and query sets used in the experiments. Afterward, we show the results of three experiments that attempt to predict alpha values. Finally, we do more comprehensive experiments on more dense retrieval models and evaluation metrics and discuss the results.

4.1 Datasets

4.1.1 Passages Set

MS MARCO Passages Dataset: **MS MARCO** (**M**icro**S**oft **M**Achine **R**eading **CO**mprehension) Bajaj et al. (2016) is a large-scale dataset focused on machine reading comprehension which consists of over 8.8 million web page passages. It is used on a large scale for benchmarking work on many information retrieval tasks. Figure 4.1 shows the length distribution of the passages in **MS MARCO Passages Dataset**, the majority of the articles are spread over a length of around 50 tokens.

4.1.2 Queries Set

- **MS MARCO dev**, which consists more than 6000 of queries. The queries in this set have, on average, only one passage that is relevant to them.

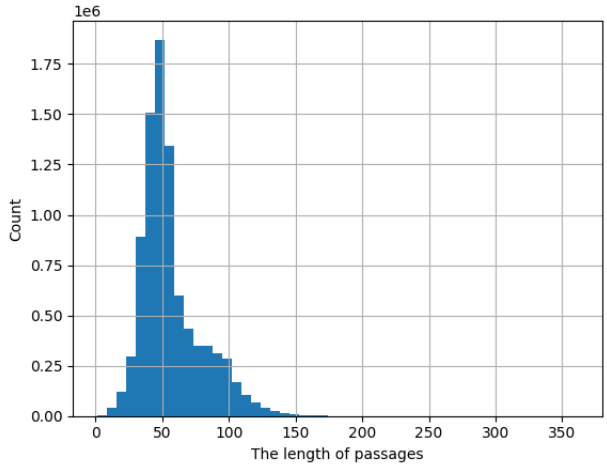


Figure 4.1: The length distribution of passages set.

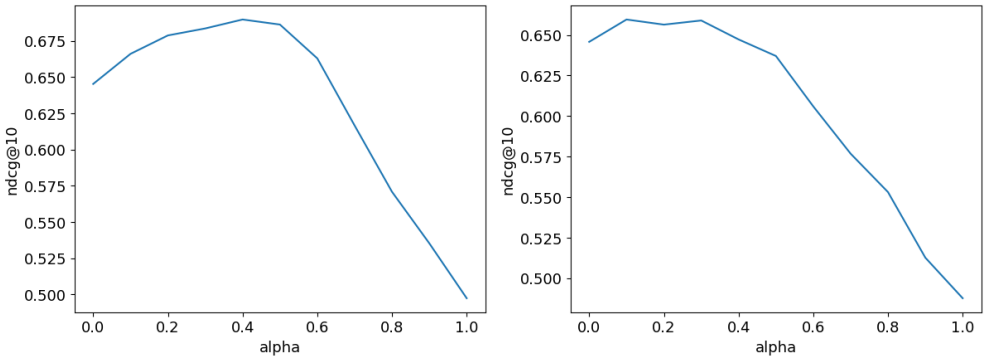
- **TREC 19**, **TREC 20**: TREC 2019,2020 Deep Learning Passage Retrieval Tasks (Craswell et al., 2020, 2021) use the same passages set as **MS MARCO dev**, but with only 43 and 54 queries,respectively. Unlike **MS MARCO dev**, each query in **TREC 19, 20** has multiple related passages and deep relevance scoring labels (relevance label: 3,2,1,0).

In the following experiments, **MS MARCO** queries set is used to train the predictors, and **TREC 19,20** queries sets are used to test the models that have already been trained. We utilize Pyserini[1] for the implementation of these models.

4.2 The Linear Interpolation Coefficients of Queries

The best nDCG@10 is obtained for the **MS MARCO Passage dev** query set when the linear interpolation coefficient $\alpha = 0.3$. But is this also the same case for other datasets? When the same linear interpolation operation is conducted on the **TREC Deep Learning Track passage retrieval task 2019** (Craswell et al., 2020) (**TREC 19**), **TREC Deep Learning Track passage retrieval task 2020** (Craswell et al., 2021) (**TREC 20**) datasets

[1]https://github.com/castorini/pyserini



(a) TREC 19 **(b) TREC 20**

Figure 4.2: TREC 19: Interpolation nDCG@10 result of BM25 and ANCE.

and the result are shown in the Figure 4.2. The figures indicate that for **TREC 19**, **TREC 20** the best nDCG@10 results are obtained when $\alpha = 0.4, 0.1$, and these are not the same as the **MS MARCO dev** dataset, which is in line with the prior results by Wang et al. (2021).

This is because the best α value is different for each query, as shown in Figure 4.3. The vast majority of queries have optimal alpha values of 0 (i.e., only using the ranked list provided by ANCE), and 1 (i.e., only using the ranked list provided by BM25), but there are some queries that do not have optimal α values of 0 and 1. It is clear from Figure 4.3 that the queries with optimal $\alpha = 1$ are the most numerous, but Figure 4.4 shows that the results are much better with $\alpha = 0$ than with $\alpha = 1$. This is because the BM25 model tends not to rank at the top, in other words, the NDCG@10 of BM25 retrieval results tends to be worse, which results in these queries with an optimal α value of 1 contributing very little to the improvement of the overall retrieval result evaluation metric. Therefore it is not feasible to apply the same α value to all queries.

Instead of taking the same α value for all queries in the linear interpolation process, the value obtained by linear interpolation using the optimal α value for each query is called the oracle value. In other words, the oracle value represents the interpolation coefficient with which the highest effectiveness (in terms of nDCG, or MAP) is achieved for an individual query. As it is shown in Figure

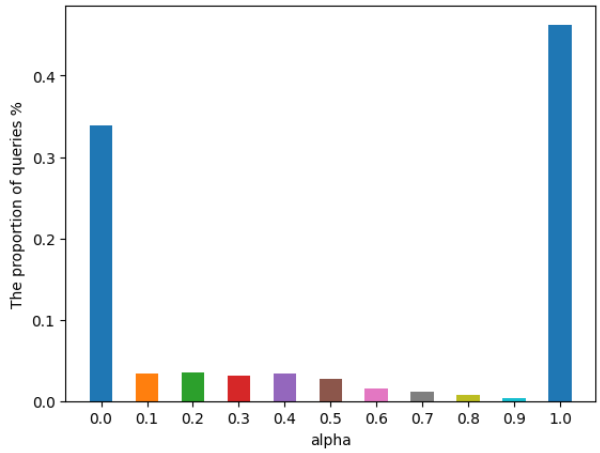


Figure 4.3: MS MARCO dev: The optimal alpha distribution for each query obtained according to the nDCG@10 value.

4.5, the red line represents the result achieved with the oracle nDCG@10 values for each individual query, and the blue line represents the results achieved with varying interpolation coefficients from 0 to 1 (same interpolation coefficient across all queries). As the gap between the two results shows, there is still a great possibility of improvement between the best effectiveness that can be achieved by the same interpolation coefficient (i.e., the blue curve) and the effectiveness achieved with the oracle values of the queries (i.e., the line in red). This is also the purpose of this report: whether a pipeline can be devised to find the optimal α value for each query to further improve the retrieval effectiveness.

To address this question, in the following, we investigate the approaches introduced in Section 3.

4.3 Only Using Query Alone As Input to Predict α

In this section, we consider whether it is possible to train a predictor which could estimate the interpolation coefficient α of a query, by only using the query itself.

To this aim, first, we use the BERT base model and only take a query

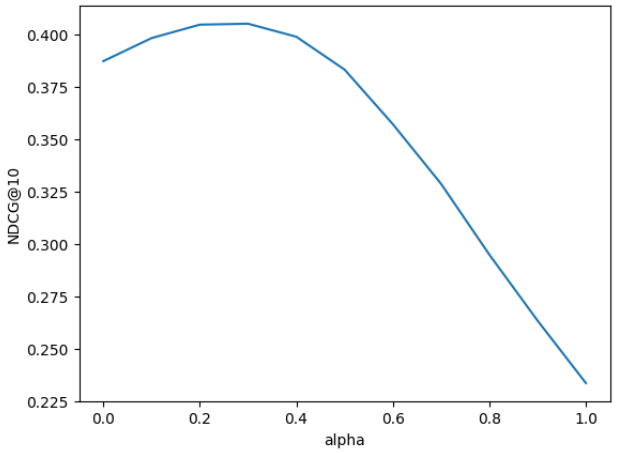


Figure 4.4: MS MARCO dev: Interpolation NDCG@10 result of BM25 and ANCE.

as input, as shown in Figure 3.3. The dataset used for training is the **MS MARCO** passages and the **dev** query set. 70% of the queries are used as the training set, 15% of the queries are used as the validation set, and 15% of the queries **TREC 19, 20** are used as the test set. They are all randomly divided according to the corresponding α value, which means they all have the same α values distribution as the whole dataset.

Two sets of experiments are conducted in this regard and the results are shown in Figure 4.6. In the first experiment, all the training data are used (green line). In the second experiment, all the queries with nDCG@10=0 are removed from the training data (yellow line). As it is shown in Figure 4.6, removing the queries with nDCG@10=0 improves the result. Although it still does not exceed the best nDCG value when $\alpha = 0.2$ (nDCG@10=0.415), it is better than using BM25 and ANCE alone (0.2471 and 0.395 respectively). 4.1 shows the prediction results of none nDCG@10=0 queries in training set on the **dev** test set, **TREC 19** and **TREC 20**. They all do not exceed the nDCG value when $\alpha = 0.2$. To further investigate the prediction results, the distribution of predicted interpolation coefficients for different queries is shown in Figure 4.7. We can see that most of the predicted values are around 0.24. Besides, Figure 4.8 shows the distribution of the ground truth interpolation

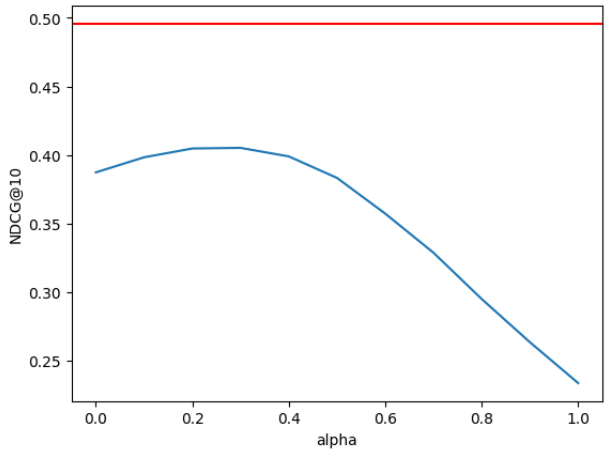


Figure 4.5: MS MARCO dev: The blue line represents the interpolation nDCG@10 result of BM25 and ANCE. The red line represents the oracle value.

	$\alpha = 0.2$	best nDCG@10	oracle nDCG@10	None nDCG@10=0
MS MARCO dev	0.4150	0.4150	0.5029	0.4091
TREC 19	0.6787	0.6897(alpha=0.4)	0.7475	0.6782
TREC 20	0.6564	0.6595(alpha=0.1)	0.7348	0.6544

Table 4.1: Query alone: The results of none nDCG@10=0 queries in training set

coefficients of the same queries, i.e., oracle values corresponding to the queries shown in Figure 4.7. Comparing the two distributions, we could infer that this method doesn't succeed in predicting the optimal α value.

4.4 Concatenating Query and Documents

The second approach that we study is to employ the top-k documents retrieved by each individual ranker to construct a ranker-specific reformulation of the query. As explained in Section 3 (Eq. 3.2), the ranker-specific reformulated queries from sparse and dense retrieval models are used as the input to the coefficient predictor. Since the average length of passages is 56 and the maximum input length of BERT is 512 tokens, the k in Eq. 3.2 is set to 5, and therefore,

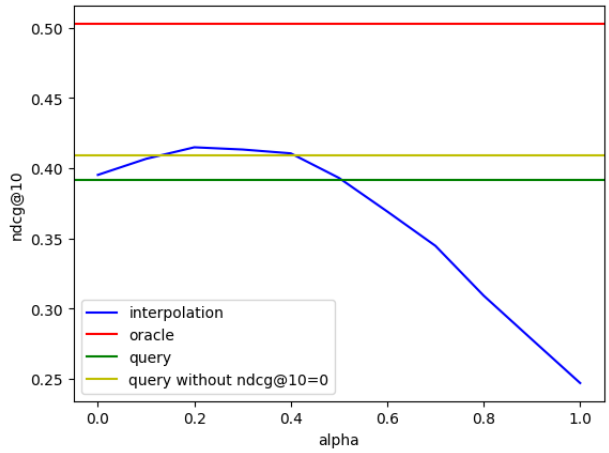


Figure 4.6: Results for the dense retriever ANCE and BM25 on the test set for varying values of interpolation parameters α (blue line). In the red line, we display the oracle value on the test set. The green line is the prediction results of the test set when using all queries in the training set to train the predictor. The green line is the prediction results of the test set when using queries without nDCG@10 = 0 in the train set to train the model.

the given query q and the top-5 passages of the BM25 and ANCE ranked lists are concatenated. The resulting concatenated query and passages are then given as the input of the cross-encoder to predict the interpolation coefficient for query q the sparse and dense retrieval models (See Figure 3.4). When the input is too long it will be automatically cut off at 512 tokens. The predictor is trained using the **MS MARCO dev** dataset, and the validation set is also part of the MS MARCO dataset. The **Trec 19** and **Trec 20** datasets are also used as test sets to test the trained models. Figure 4.9 shows the interpolation results of the **TREC 19** and **TREC 20** dataset, the best nDCG@10 values are obtained when $\alpha = 0.4$ and $\alpha = 0.1$, which are different with **MS MARCO dev**. The distribution of α labels in the training set is very biased, with most of them being 0 and 1, so three cases are considered:

- removing all queries with $\alpha = 0, 1$ in the training set
- keeping all queries with $\alpha = 0, 1$ in the training set
- sampling a part of the queries with $\alpha = 0, 1$ in the training set, the final

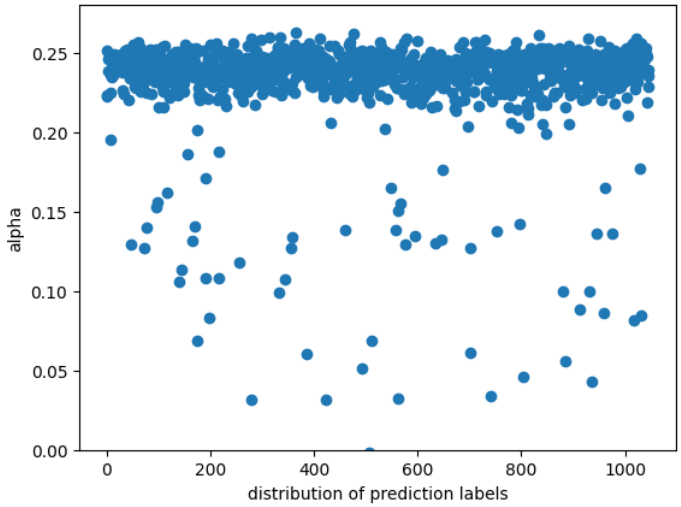


Figure 4.7: Query alone: Prediction results (α) distribution.

α	0.2	0.1	0.4	0.3	0	1	0.5	0.6	0.7	0.8	0.9
Proportion	0.136	0.131	0.128	0.119	0.118	0.118	0.106	0.060	0.043	0.027	0.015

Table 4.2: The ratio of each α value of the sample training data

ratio of each alpha is shown in the table 4.2

The result is shown in the table 4.3. $\alpha = 0.2$ is the optimal α value of validation set. Training sets with different distributions will lead to different training results, and I have looked further into the prediction distributions. Figure A.1 shows the distribution of prediction results for different test sets under different conditions. In the case of removing all the queries with α=0, 1, all the prediction results are very concentrated and distributed around 0.22. In the case of keeping all the queries with α=0, 1, all the prediction results are very concentrated and distributed around 0.4. Things change a lot when sampling a part of the queries. Although the nDCG@10 result at this time is not the best, the distribution of predicted values becomes less concentrated.

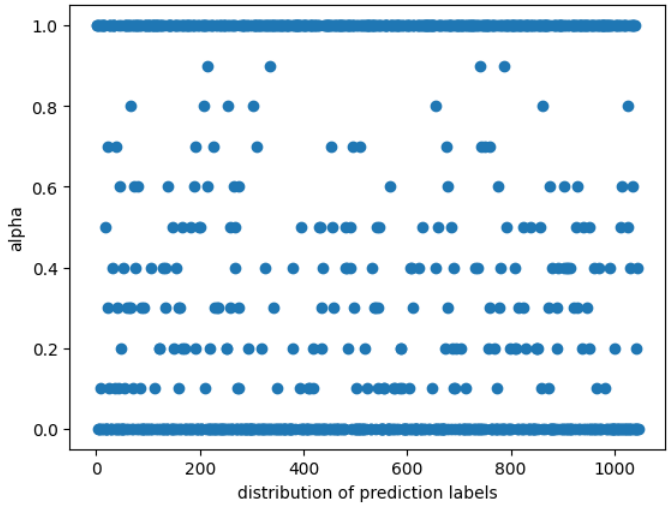
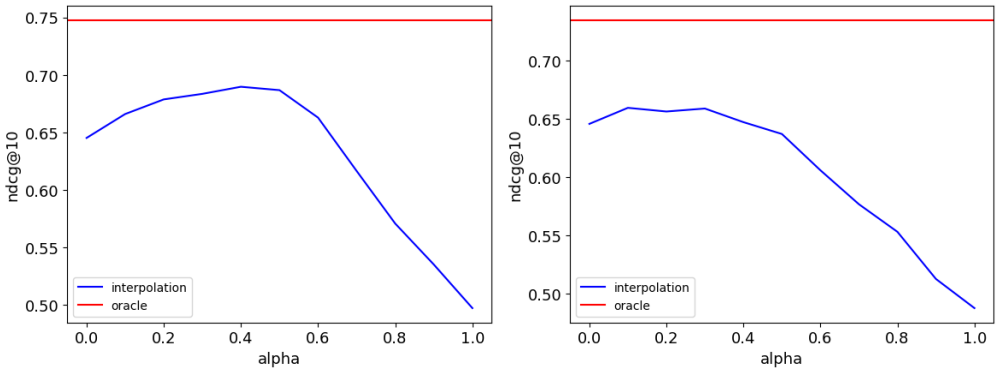


Figure 4.8: Distribution of oracle interpolation coefficients.

	$\alpha = 0.2$	best nDCG@10	oracle nDCG@10	None α=0,1	all α=0,1	balance α
MS MARCO dev	0.4150	0.4150	0.5029	0.3968	0.4002	0.3981
TREC 19	0.6787	0.6897(alpha=0.4)	0.7475	0.6940	0.6768	0.6913
TREC 20	0.6564	0.6595(alpha=0.1)	0.7348	0.6466	0.6579	0.6484

Table 4.3: cross encoder: The results of alpha=0, 1 in different proportions



(a) **(b)**

Figure 4.9: Results for the dense retriever ANCE and BM25 on **TREC 19** (Figure 4.9a) and **TREC 20** (Figure 4.9b) for varying values of interpolation parameters α (blue line). In the red line, I display the oracle value.

	$\alpha = 0.2$	best nDCG@10	oracle nDCG@10	10 tokens	20 tokens	30 tokens
MS MARCO dev	0.4150	0.4150	0.5029	0.3921	0.3919	0.3887
TREC 19	0.6787	0.6897(alpha=0.4)	0.7475	0.6964	0.6953	0.6920
TREC 20	0.6564	0.6595(alpha=0.1)	0.7348	0.6556	0.6569	0.6515

Table 4.4: The result of the query expansion: 10, 20, and 30 tokens are expanded after the original query.

4.5 Query Expansion

In this part of the experiments, we use the RM3 relevance model to expand the queries as the input of the cross-encoder to predict the interpolation coefficient for query q and only consider the case of sampling a part of the queries with relatively balanced α. Distilroberta-base (Sanh et al., 2019) is used as the head of the cross-encoder. The expanded form of the query is shown in the equation 3.3. The number of feedback passages is set at 10 and the number of feedback terms k in equation 3.3 is set to 10,20,30. The input data form is:

$$[CLS]\text{expanded query}_{ANCE}[SEP]\text{expanded query}_{BM25} \tag{4.1}$$

The results are shown in table 4.4. Compared with the result of concatenating queries and documents, the result of query expansion is slightly better but the computing resources required become less. As the number of added tokens increases, the results of nDCG@10 become slightly worse. For the **TREC 19** dataset, the predicted result (nDCG@10=0.6964) is a little better than when α=2 (nDCG@10=0.6787). Figure A.1 shows the distribution of prediction results for different test sets by adding a different number of tokens. We can see that increasing the number of tokens has little effect on the distribution of prediction results.

4.6 Deep Evaluation Metric

Previous experiments have considered nDCG@10 as the evaluation metric, which is a shallow evaluation. As shown in the section 4.5, by combining the original query with information from BM25 and the ANCE retrieval results using query expansion, the nDCG@10 prediction results obtained by feeding the newly generated query pairs into the cross-coder have gained some improvement, compared to using the same alpha globally. In the following experiment, we use the same experimental procedure of cross-encoder but with two deep evaluation metrics: MAP and nDCG@1000.

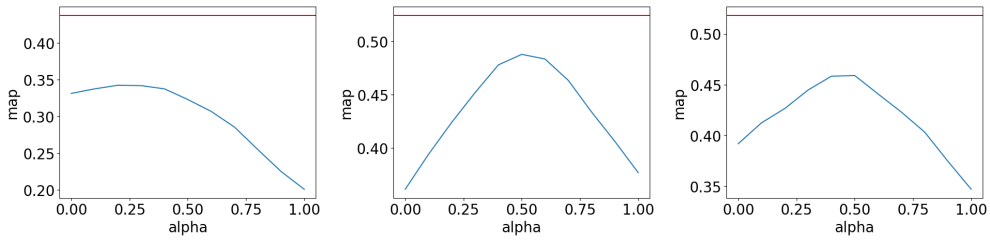
4.6.1 MAP

As shown in Figure 4.10, The optimal α value of **MS MARCO dev** test set becomes 0.2, and the optimal α value of **Trec 19**, **Trec 20** query sets becomes 0.5, which are different from the results of nDCG@10. Also, the linear interpolation of BM25 and ANCE retrieval provides a somewhat greater improvement in the deep evaluation metric than the shallow evaluation metric. The reason why the interpolation result of the **MS MARCO dev** set is not affected by the shallow or deep evaluation is that the queries in this set on average associate to only one relevant passage.

The experiment uses 70% of the **MS MARCO dev** query set to train the cross-encoder and 15% of the **MS MARCO dev** query set as the validation set. The test sets are the remaining 15% **MS MARCO dev** query set, **TREC 19** query set, and **TREC 20** query set. The number of expanded terms k in equation 3.3 is set to 10. The result is shown in table 4.5. For **TREC 19** and **Trec 20**, the prediction results are better than both the best MAP value and MAP value of $\alpha = 0.2$ (the optimal alpha from the validation set). Besides, compared with the results of nDCG@10 (only get improvement on **TREC 19**), the trained model has a better performance.

	$\alpha = 0.2$	best map	oracle map	our method
MS MARCO dev	0.3511	0.3511(alpha=0.2)	0.4378	0.3320
TREC 19	0.4241	0.4879(alpha=0.5)	0.5246	0.4903
TREC 20	0.4266	0.4590(alpha=0.5)	0.5181	0.4615

Table 4.5: The results of MAP evaluation metric.



(a) **(b)** **(c)**

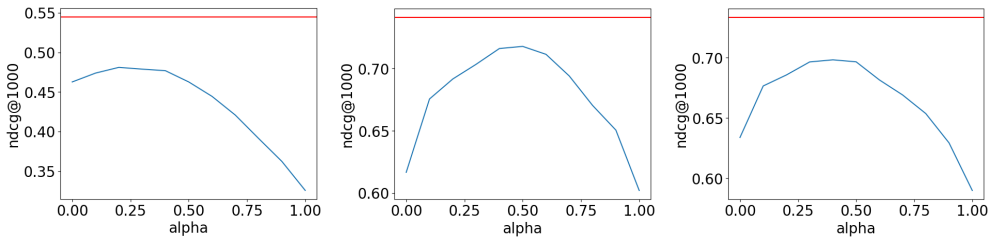
Figure 4.10: MAP results for the dense retriever ANCE and BM25 on (4.10a)**MS MARCO dev** test set, (4.10b)**Trec 19**, and (4.10c)**Trec 20** for varying values of interpolation parameters α (blue line). In the red line, I display the oracle value.

4.6.2 nDCG@1000

Next, we consider using nDCG@1000 as the evaluation metric. Like the case of MAP, the optimal alpha values (labels) for queries become somewhat different when using nDCG@1000 compared to using nDCG@10 as an evaluation metric.

As shown in Figure 4.11a, the nDCG@1000 results of the **MS MARCO dev** query set are very similar to the nDCG@10 results, and the best nDCG@1000 value is obtained when α=0.2. But the nDCG@1000 results of the **TREC 19** and **TREC 20** datasets are quite different from the nDCG@10 results. As shown in the Figure 4.11b, 4.11c, for the **TREC 19** dataset, the best α value changes from 0.4 to 0.5, and for the **TREC 20** dataset, the best α value changes from 0.1 to 0.4. This is similar to the results of previous MAP experiments.

The experiment uses 70% of the **MS MARCO dev** query set to train the cross-encoder and 15% of the **MS MARCO dev** query set as the validation



(a) **(b)** **(c)**

Figure 4.11: nDCG@1000 results for the dense retriever ANCE and BM25 on (4.11a)**MS MARCO dev** test set, (4.11b)**Trec 19**, and (4.11c)**Trec 20** for varying values of interpolation parameters α (blue line). In the red line, I display the oracle value.

set. The test sets are the remaining 15% **MS MARCO dev** query set, **TREC 19** query set, and **TREC 20** query set. The number of expanded terms k from each of the BM25 and ANCE ranking lists in equation 3.3 is set to 10. The result is shown in table 4.6. The prediction result of both **TREC 19** is a little better than the best nDCG@1000 and nDCG@1000 of alpha=0.2 (the optimal alpha of the validation set). The prediction result of **TREC 20** is only a little better than the nDCG@1000 of $\alpha = 0.2$ (the optimal alpha of the validation set).

	alpha = 0.2	best nDCG@1000	oracle nDCG@1000	our method
MS MARCO dev	0.4810	0.4810(alpha=0.2)	0.5447	0.4608
TREC 19	0.6914	0.7176(alpha=0.5)	0.7409	0.7182
TREC 20	0.6857	0.6983(alpha=0.4)	0.7336	0.6959

Table 4.6: The results of nDCG@1000 evaluation metric.

4.7 More Dense Retrieval Models And Evaluation Metrics

It is clear from the results of the experiments in table 4.5 and 4.6 that different depths of evaluation metrics can have a significant impact on the final results. Therefore, in this part of the experiments, we conduct more comprehensive experiments to explore the effects of interpolation with different dense retrieval models and different depths of evaluation metrics on the results. Three other dense retrieval models (SBERT, TCT-ColBERTv2, DistilBERT KD) in addition to ANCE are used for linear interpolation with BM25 and nDCG@10,100,100, MAP are taken into account for the following experiments.

At first, we compare the results of our method with the cases that $\alpha = 0.5$ (dense and sparse linear interpolation weights are equal) and α from the validation set on four different dense retrieval models, and the results are shown in table 4.7 4.8 A.1 A.2. Paired samples statistical significance t-tests are conducted between our method and the case α from the validation set and significant differences are marked as bold. In the cases of MAP, nDCG@100, and nDCG@1000 as evaluation metrics, our method outperforms the results of the α from the validation set. However, in the case of nDCG@10, our method performed relatively poorly. Therefore, it demonstrates the stronger performance of our method on the deeper evaluation metrics than on the shallow evaluation metrics.

Then, we compare the results of our method with the retrieval results of BM25 and dense retrieval models respectively, and the results are shown in table 4.7 4.8 A.1 A.2. Paired samples statistical significance t-tests are conducted between our method and BM25 and dense retrieval results and significant differences are marked with † and ‡. In the case of all MAP, nDCG@100, and nDCG@1000, our method significantly outperforms the results of BM25 and dense retrieval in all combinations of linear interpolation. However, the results at nDCG@10 are not as good, especially for the linear interpolation of BM25 and TCT-ColBERTv2, our method is even worse than the results of TCT-ColBERTv2 alone retrieval on the **TREC 20** query set.

dense	BM25 results		dense retrieval results		α=0.5		α from validation		our method	
	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20
ANCE	0.3768	0.3472	0.3611	0.3919	0.4880	04590	0.4241(0.2)	0.4266(0.2)	**0.4903**$^{\dagger}_{\ddagger}$	**0.4615**$^{\dagger}_{\ddagger}$
TCTv2	0.3768	0.3472	0.4452	0.4514	0.5224	0.4921	0.4933(0.2)	0.4869(0.2)	0.5285$^{\dagger}_{\ddagger}$	0.4944$^{\dagger}_{\ddagger}$
KD	0.3768	0.3472	0.3759	0.3909	0.5701	0.4888	0.4635(0.3)	0.4603(0.3)	**0.5053**$^{\dagger}_{\ddagger}$	**0.4799**$^{\dagger}_{\ddagger}$
SBERT	0.3768	0.3472	0.4097	0.3990	04971	0.4534	0.4595(0.2)	0.4342(0.2)	**0.5050**$^{\dagger}_{\ddagger}$	**0.4597**$^{\dagger}_{\ddagger}$

Table 4.7: The MAP results of BM25, dense retrieval models, and all interpolation runs of all sparse and dense retrieval models combinations. In parentheses is the α value corresponding to the evaluating metric value. Statistical significance tests are conducted between our method and α from the validation set, significant differences are marked as bold ($p < 0.05$). Statistical significance tests are also conducted between our method and BM25, dense retrieval models, significant differences are marked with † and ‡ ($p < 0.05$).

4.8 Other Validation Experiments

In this section, we conducted two verification experiments. Since in previous experiments, we have been using the validation set to obtain the best settings for the model parameters, in the first experiment we aim to verify the effect of early stop on the results. In the second experiment, we aim to verify whether a larger training set would have a better impact.

4.8.1 Early Stop

The model parameter settings obtained at the 10, 20,30, 40, and 50 epochs of the training process are selected and compared with those selected from the validation set. In this part of the experiment, we use the interpolated combination of BM25 and ANCE and query expansion method. Figure 4.12 shows the change in the prediction results distribution as the training epochs increase. The optimal model selected from the validation set is between 10 and 20 epochs. Table 4.9 shows the MAP results of **TREC 19** and **TREC 20** corresponding to different epochs. For **TREC19**, the validation set selects the

dense	BM25 results		dense retrieval results		α=0.5		α from validation		our method	
	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20
ANCE	0.4973	0.4876	0.6452	0.6458	0.6862	0.6370	0.6787(0.2)	0.6564(0.2)	$0.6964^{\dagger}_{\ddagger}$	$0.6556^{\dagger}$
TCTv2	0.4973	0.4876	0.6867	0.6950	0.6934	0.6672	0.7142(0.2)	0.71(0.2)	$0.7117^{\dagger}$	0.6782
KD	0.4973	0.4876	0.6994	0.6447	0.7135	0.6788	0.7225(0.3)	0.6987(0.3)	$0.7300^{\dagger}$	$0.7074^{\dagger}_{\ddagger}$
SBERT	0.4973	0.4876	0.6930	0.6344	0.6888	0.6242	0.7073(0.3)	0.6401(0.3)	$0.7014^{\dagger}$	$0.6441^{\dagger}$

Table 4.8: The nDCG@10 results of BM25, dense retrieval models, and all interpolation runs of all sparse and dense retrieval models combinations. In parentheses is the α value corresponding to the evaluating metric value. Statistical significance tests are conducted between our method and α from the validation set, significant differences are marked as bold ($p < 0.05$). Statistical significance tests are also conducted between our method and BM25, dense retrieval models, significant differences are marked with † and ‡ ($p < 0.05$).

	10 epochs	15 epochs	20 epochs	30 epochs	40 epochs	50 epochs	validation
TREC19	0.4844	0.4842	0.4882	0.4600	0.4468	0.4399	**0.4903**
TREC20	0.4619	0.4572	**0.4622**	0.4491	0.4283	0.4426	0.4615

Table 4.9: MAP results of different epochs.

best model. And for **TREC 20**, the best result is obtained at 20 epochs, but the difference between them is small. In general, although the validation set is also based on sparse relevance judgments, the optimal model settings can still be relatively selected.

4.8.2 Large Training Set

In this section, we use a part of **MS MARCO training query set** to train the predictor. The query set contains 50,000 queries, and after removing part of the queries with α=0, 1 to make it not biased, the final number of queries used in training is about 30,000 (previously it is about 3,000 queries).
Results using large and small training sets are shown in Table 4.10. There is

	best same α for all queries	$\alpha = 0.5$	α from validation set	small training set	large training set
TREC19	0.4880(0.5)	0.4880(0.5)	0.4520(0.3)	0.4903	0.4840
TREC20	0.4590(0.5)	0.4590(0.5)	0.4448(0.3)	0.4615	0.4653

Table 4.10: The MAP results of BM25+ANCE linear interpolation.

little difference in the training results using different sizes of data sets, which may be because the queries in both the **MS MARCO dev** and **MS MARCO training set** on average associate to only one relevant passage.

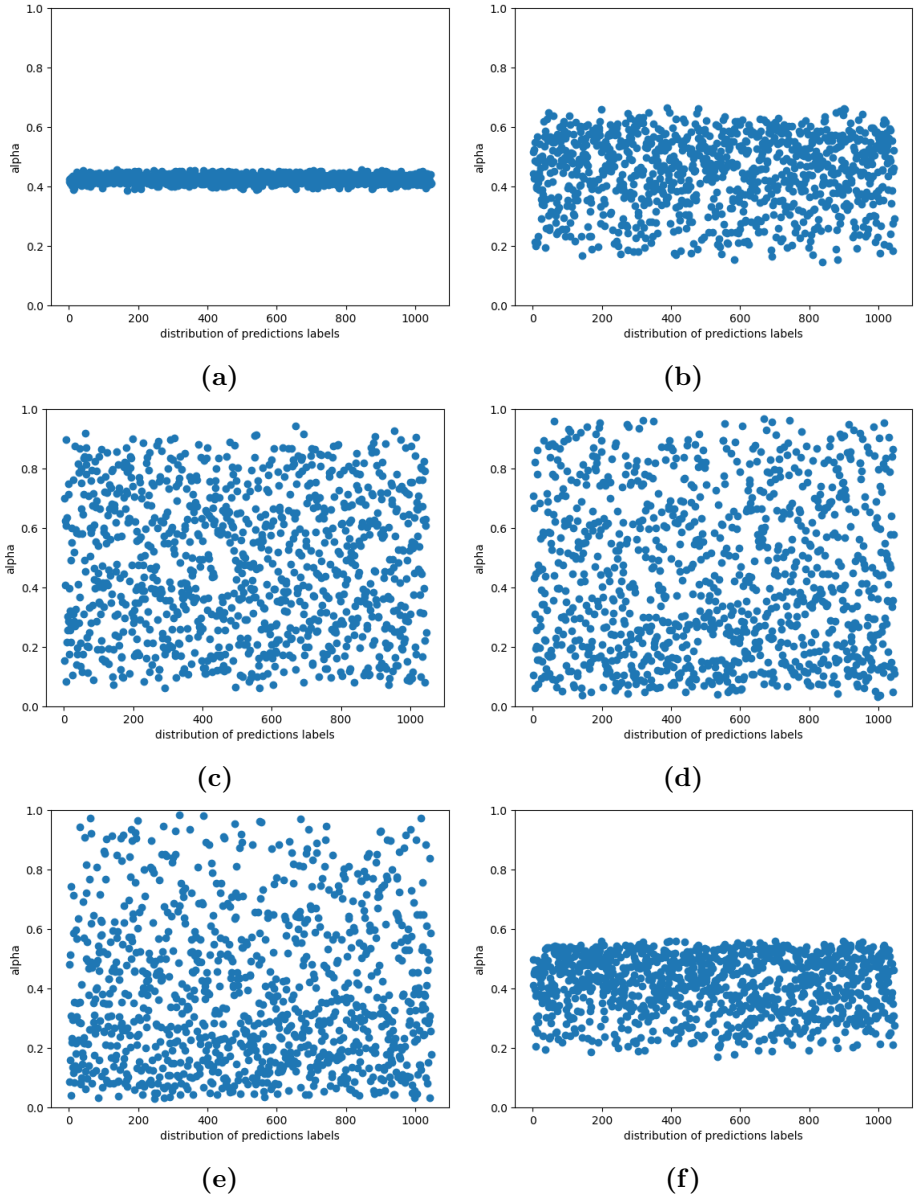


Figure 4.12: The distribution of prediction results of the test set (15% of MS MARCO dev): 4.12a, 4.12b, 4.12c, 4.12d, 4.12e are the results of 10, 20, 30, 40, 50 epochs training respectively(without validation set). 4.12f is the prediction result of the model selected by the validation set.

Chapter 5

Discussion

Linear interpolation of sparse and dense retrieval can improve retrieval performance. By performing BM25 and ANCE linear interpolation separately for each query I find that the optimal linear interpolation coefficients α are not always the same for each query, which led to us not being able to use the same α for a collection of queries. I devised some experiments to explore a pipeline that would predict the linear interpolation coefficients relatively successfully.

5.1 RQ1: How to effectively estimate the interpolation coefficient in hybrid ranking with sparse and dense retrieval models?

First, I used only queries as input to the model to predict the α. The results show a complete failure to predict the α, which is expected. Then I try to directly concatenate the query and the top 5 documents retrieved from the ranked list of the sparse and dense search models as input to the cross-encoder to predict the α. The trained model doesn't result in better prediction performance, but through further exploration of the training data labels, I find that reducing the percentage of $\alpha = 0,1$ to no bias could appropriately improve the training results. The reason why the direct concatenating of documents and queries does not work well is probably that the length of the document

being concatenated is much longer than the length of the query, which causes more attention to be focused on the passages rather than the concatenated sentences of the query and passages. Then, I try to use the queries expanded by the RM3 relevance model (add ten tokens after each query) as input to the cross-encoder. The prediction results of the trained model on the **TREC 19** and **TREC 20** show a very significant improvement.

5.2 RQ2: How do the results of interpolation coefficient estimation vary for different evaluation metrics?

As the evaluation metrics used in the previous experiments are all nDCG@10 which is a kind of shallow evaluation metric, I next consider using deep evaluation metrics: MAP and nDCG@1000. I find that each query has different optimal α under different evaluation metrics, and the results of the models trained using them are different. The results are somewhat better when using the deeper evaluation metrics than nDCG@10 (the predictions are better than the case of the α from the validation set on the **TREC 19 TREC 20** set).

5.3 RQ3: How do the results of interpolation coefficient estimation vary across different sparse and dense retrieval models?

More dense retrieval models (ANCE, TCT-ColBERTv2, DistilBERT KD, SBERT) and evaluation metrics (MAP, nDCG@10, 100, 1000) are considered. Comparing the prediction results of the trained model with the case of the α from the validation set, some improvement is also obtained, which shows that our method is effective not only on the ANCE model. Similarly, linear interpolations with models other than ANCE also perform better in the deep evaluation metrics. Comparing the predictions of our method with the BM25 and dense retrieval results respectively, our method is better than both of them (except in

the case of the linear interpolation with TCT-ColBERTv2, which is probably because its own retrieval performance is already very good, which leads to that there are no great possibilities for improvement), especially in the case of the deep evaluation metrics. I then considered whether setting a fixed number of early stop steps instead of using a validation set would give an improvement, and it turns out that using a validation set still worked better. At last, I use a larger training set to train the model and the results are similar to those of the smaller training set.

5.4 Limitation and Future Work

A major limitation of our method is that the queries in the training set have on average only one relevant passage, but the queries in the **TREC 19** and **TREC 20** test sets have much deeper relevance judgments pools. This is the reason why there is little difference in the results using a larger training set. In future research, using queries with more relevant passages to train the models is also worth being considered. Another limitation is that there are not enough GPU resources, all experiments use the same set of passages and only the set of queries is constantly being changed. In the later study, it can be verified whether our method has the same improvement on other datasets.

Chapter 6

Conclusion

In conclusion, the study has shown that linear interpolation of sparse and dense retrieval can improve retrieval performance. The optimal linear interpolation coefficients α are not always the same for each query, and predicting them using only queries or directly concatenating queries and documents proved to be unsuccessful. However, using queries expanded by the RM3 relevance model as input to the cross-encoder significantly improved prediction results, especially when using deep evaluation metrics like MAP and nDCG@1000. The method was found to be effective not only on the ANCE model but also on other dense retrieval models, and it outperformed both BM25 and dense retrieval results in most cases, especially in deep evaluation metrics. However, the study's limitations include the lack of relevant passages in the training set queries and the need for more GPU resources for further experimentation. Future research can explore using queries with more relevant passages to train the models and verify the method's improvement on other datasets.

Appendix A

dense	BM25 results		dense retrieval results		α=0.5		α from validation		our method	
	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20
ANCE	0.4981	0.4914	0.5540	0.5679	0.6333	0.6173	0.5895(0.2)	0.5933(0.2)	$0.6347^{\dagger}_{\ddagger}$	$\mathbf{0.6188}^{\dagger}_{\ddagger}$
TCTv2	0.4981	0.4914	0.6129	0.6200	0.6545	0.6447	0.6429(0.2)	0.6442(0.2)	$0.6631^{\dagger}_{\ddagger}$	$\mathbf{0.6569}^{\dagger}_{\ddagger}$
KD	0.4981	0.4914	0.5765	0.5728	0.6516	0.6494	0.6475(0.4)	0.6456(0.4)	$0.6515^{\dagger}_{\ddagger}$	$0.6481^{\dagger}_{\ddagger}$
SBERT	0.4981	0.4914	0.5985	0.5734	0.6382	0.6080	0.6207(0.2)	0.5967(0.2)	$\mathbf{0.6493}^{\dagger}_{\ddagger}$	$\mathbf{0.6184}^{\dagger}_{\ddagger}$

Table A.1: The NDCG@100 results of BM25, dense retrieval models, and all interpolation runs of all sparse and dense retrieval models combinations. In parentheses is the α value corresponding to the evaluating metric value. Statistical significance tests are conducted between our method and α from the validation set, significant differences are marked as bold ($p < 0.05$). Statistical significance tests are also conducted between our method and BM25, dense retrieval models, significant differences are marked with † and ‡ ($p < 0.05$).

dense	BM25 results		dense retrieval results		α=0.5		α from validation		our method	
	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20	TREC19	TREC20
ANCE	0.6001	0.5874	0.6165	0.6337	0.7171	0.6966	0.6914(0.2)	0.6857(0.2)	$0.7180^{\dagger}_{\ddagger}$	$0.69834^{\dagger}_{\ddagger}$
TCTv2	0.6001	0.5874	0.6929	0.6906	0.7392	0.7240	0.7355(0.2)	0.7297(0.2)	$0.7451^{\dagger}_{\ddagger}$	$0.7328^{\dagger}_{\ddagger}$
KD	0.6001	0.5874	0.6421	0.6405	0.7316	0.7243	0.7316(0.4)	0.7254(0.4)	$0.7329^{\dagger}_{\ddagger}$	$0.7257^{\dagger}_{\ddagger}$
SBERT	0.6001	0.5874	0.6687	0.6421	0.7254	0.6947	0.7194(0.2)	0.6933(0.2)	$0.7323^{\dagger}_{\ddagger}$	$0.6994^{\dagger}_{\ddagger}$

Table A.2: The NDCG@1000 results of BM25, dense retrieval models, and all interpolation runs of all sparse and dense retrieval models combinations. In parentheses is the α value corresponding to the evaluating metric value. Statistical significance tests are conducted between our method and α from the validation set, significant differences are marked as bold ($p < 0.05$). Statistical significance tests are also conducted between our method and BM25, dense retrieval models, significant differences are marked with † and ‡ ($p < 0.05$).

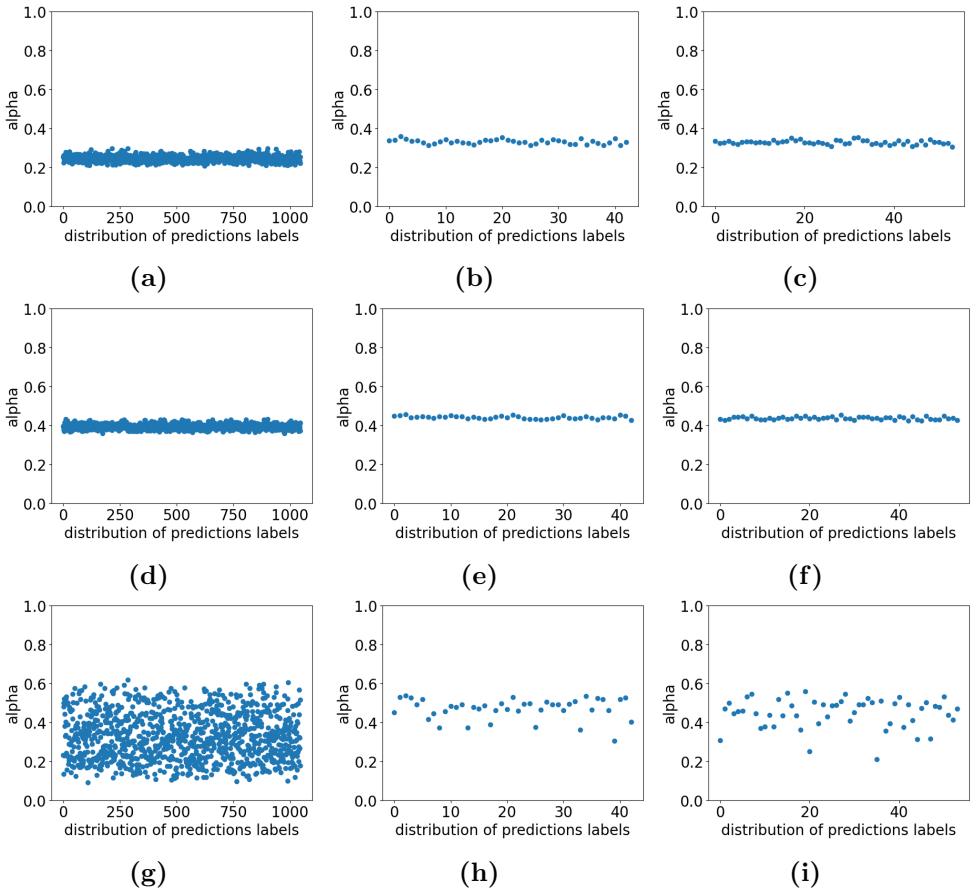


Figure A.1: The distribution of prediction results of the test set: the first column is MS MARCO dev, the second column is **TREC 19**, and the third column is **TREC 20**. The results in the first row are the case that the training set containing all α=0,1 queries. The results in the second row are the case that the training set does not contain alpha=0, 1. The results in the third row are the case that the training set with alpha=0, 1 query with the balanced distribution.

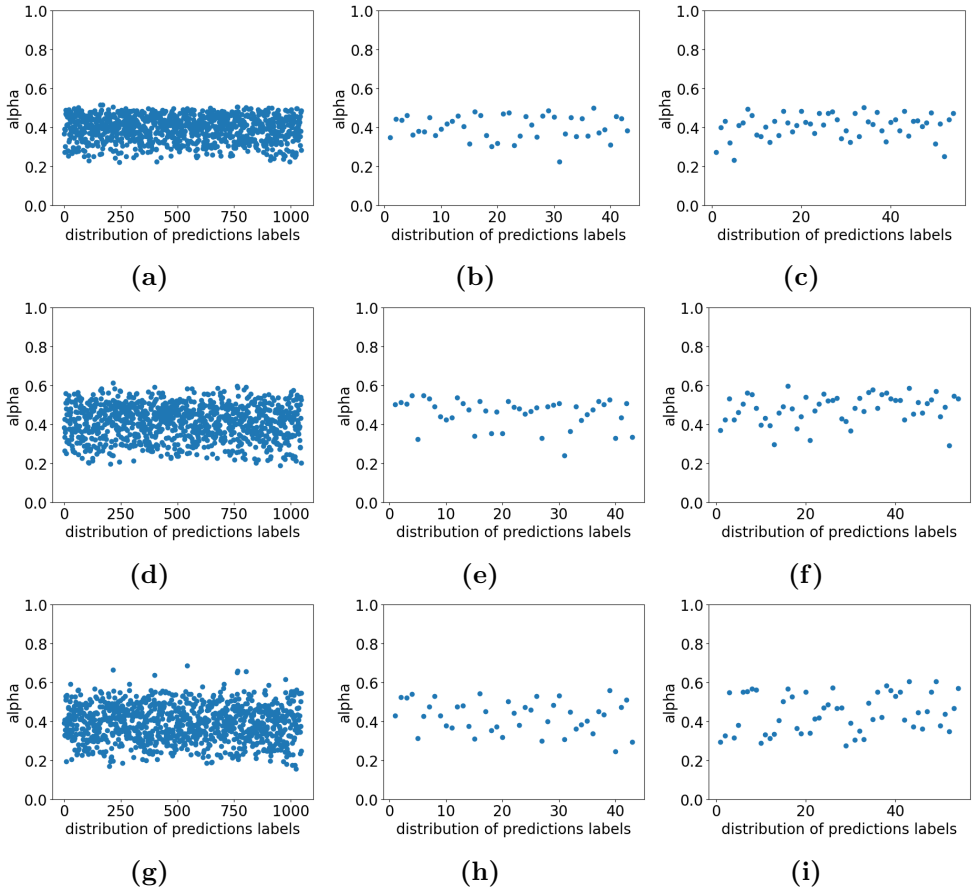


Figure A.2: The distribution of prediction results of the test set: the first column is **MS MARCO dev**, the second column is **TREC 19**, and the third column is **TREC 20**. The model in the first row uses queries extended by 10 tokens. The model in the second row uses queries extended by 20 tokens. The model in the third row uses queries extended by 30 tokens.

Bibliography

[1] *SIGIR '22: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323.

[2] N. Abduljaleel, J. Allan, W. B. Croft, O. Diaz, L. Larkey, X. Li, D. Metzler, MD Smucker, T. Strohman, and H. Turtle. Umass at trec 2004: Notebook. *Trec*, pages 657–670, 2004.

[3] Arian Askari, Amin Abolghasemi, Gabriella Pasi, Wessel Kraaij, and Suzan Verberne. Injecting the bm25 score as text improves bert-based re-rankers, 2023.

[4] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset, 2016. URL https://arxiv.org/abs/1611.09268.

[5] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055, 2017. URL http://arxiv.org/abs/1708.00055.

[6] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the trec 2019 deep learning track, 2020.

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2020 deep learning track, 2021.

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. Overview of the trec 2022 deep learning track. In *Text REtrieval Conference (TREC)*. NIST, TREC, March 2023.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[10] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, and Jiafeng Guo. Pre-training methods in information retrieval, 2022.

[11] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation, 2021.

[12] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.

[13] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020.

[14] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 120–127, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133316. doi: 10.1145/383952.383972. URL https://doi.org/10.1145/383952.383972.

[15] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls, 2022.

[16] Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon. To interpolate or not to interpolate: Prf, dense and sparse retrievers. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2495–2500, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531884. URL https://doi.org/10.1145/3477495.3531884.

[17] Jimmy Lin and Xueguang Ma. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques, 2021.

[18] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362, 2021.

[19] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: Bert and beyond, 2021.

[20] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling dense representations for ranking using tightly-coupled teachers, 2020.

[21] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.repl4nlp-1.17. URL https://aclanthology.org/2021.repl4nlp-1.17.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[23] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 1895–1898, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585123. doi: 10.1145/1645953.1646259. URL https://doi.org/10.1145/1645953.1646259.

[24] Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, page 579–586, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450301534. doi: 10.1145/1835449.1835546. URL https://doi.org/10.1145/1835449.1835546.

[25] Craig Macdonald and Nicola Tonellotto. Declarative experimentation ininformation retrieval using pyterrier. In *Proceedings of ICTIR 2020*, 2020.

[26] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[27] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 01 2009. doi: 10.1561/1500000019.

[28] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[29] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. doi: 10.1109/cvpr.2015.7298682. URL https://doi.org/10.1109%2Fcvpr.2015.7298682.

[30] Olga Vechtomova and Ying Wang. A study of the effect of term proximity on query expansion. *Journal of Information Science*, 32(4):324–333, 2006.

[31] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21, page 317–324, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450386111. doi: 10.1145/3471158.3472233. URL https://doi.org/10.1145/3471158.3472233.

[32] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020.

[33] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey, 2022. URL https://arxiv.org/abs/2211.14876.

[34] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey, 2022.

[35] Shengyao Zhuang and Guido Zuccon. Tilde: Term independent likelihood model for passage re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1483–1492, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3462922. URL https://doi.org/10.1145/3404835.3462922.